ENGR 451 - Chapter 2 Laboratory

Matlab tutorial

```
clear
x = sequence([1 2 3 4 5], 1);
y = sequence([5 3 1 -1 3 -2 2 3], -1);

% test plus
test_lab1('plus(x, y)')
test_lab1('plus(y, x)')
test_lab1('plus(1, x)')
test_lab1('plus(x, 1)')

y = sequence([5 3 1 0 3 -2 2 3], -4);
test_lab1('plus(x, y)')
test_lab1('plus(y, x)')

% test minustract
test_lab1('minus(x, y)')
test_lab1('minus(y, x)')
test_lab1('minus(1, x)')
test_lab1('minus(x, 1)')

% test timesiplication
test_lab1('times(x, y)')
test_lab1('times(3, x)')
test_lab1('times(x, 3)')

% test flip
test_lab1('flip(x)')

% test shift
test_lab1('shift(y, 2)')

%combinations
test_lab1('flip(minus(shift(plus(x, 2), 4), y))')
test_lab1('plus(flip(plus(x, y)), shift(y, -5))')
test_lab1('minus(plus(times(shift(flip(x), 4), shift(y, 3)), flip(y)), x)')

% test stem
set(clf, 'Position', [200 200 400 200])
stem(flip(2+(x-shift(y, -4).*y-3)))
title('y[n]');

% Program Listings
fprintf('\n\n')
disp('--- sequence.m --------------------------------')
type sequence
```

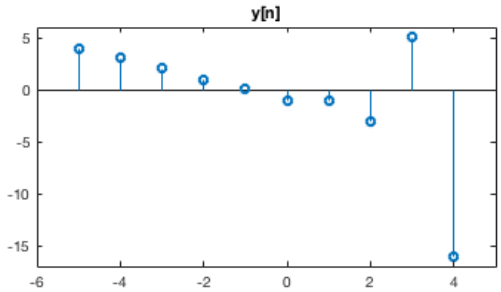
```
plus(x, y): sequence O.K.
plus(y, x): sequence O.K.
plus(1, x): sequence O.K.
plus(x, 1): sequence O.K.
plus(x, y): sequence O.K.
plus(y, x): sequence O.K.
minus(x, y): sequence O.K.
minus(y, x): sequence O.K.
minus(1, x): sequence O.K.
minus(x, 1): sequence O.K.
times(x, y): sequence O.K.
times(3, x): sequence O.K.
times(x, 3): sequence O.K.
flip(x): sequence O.K.
shift(y, 2): sequence O.K.
flip(minus(shift(plus(x, 2), 4), y)): sequence O.K.
plus(flip(plus(x, y)), shift(y, -5)): sequence O.K.
minus(plus(times(shift(flip(x), 4), shift(y, 3)), flip(y)), x): sequence O.K.


--- sequence.m --------------------------------

classdef sequence
        properties
                data
                offset
        end

        methods
                function s = sequence(data, offset)
                        % SEQUENCE   Sequence object
                        %            S = SEQUENCE(DATA, OFFSET) creates sequence S
                        %            using DATA and OFFSET
                        %
```

```
            %            Cindy Phan 13 Feb 2019
            s.data = data;
            s.offset = offset;
    end

    function display(s)
            var = inputname(1);
            if (isempty(var))
                    disp('ans =');
            else
                    disp([var '=']);
            end
            switch length(s.data)
                    case 0
                            disp('    data: []')
                    case 1
                            disp(['    data: ', num2str(s.data)])
                    otherwise
                            disp(['    data: [' num2str(s.data) ']'])
            end
            disp(['  offset: ' num2str(s.offset)])
    end

    function y = flip(x)
            % FLIP Flip a Matlab sequence structure, x, so y = x[-n]
f = fliplr(x.data);
x.data = f;
f1 = -(x.offset)-(length(x.data)-1);
x.offset = f1;
y = x;
    end

    function y = shift(x, n0)
            % SHIFT Shift a Matlab sequence structure, x, by integer amount n0 so that y[n] = x[n - n0]
s1 = x.offset+(n0);
x.offset = s1;
y = x;
    end

    function z = plus(x, y)
            % PLUS  Add x and y. Either x and y will both be sequence structures, or one of them may be a number.

%Check to see if either x or y are an integer, and make an
%array if necessary

if isa(x,'sequence')==0;
p1 = sequence((x)*(ones(1,length(y.data))),y.offset);
x = p1;
end

if isa(y,'sequence')==0;
p2 = sequence((y)*(ones(1,length(x.data))),x.offset);
y = p2;
end

%concatenate zeros in front of x.data or y.data if necessary

pxo=x.data; %save original x.data & y.data for length usage
pyo=y.data;

if x.offset>y.offset;
    px3 = zeros((x.offset-y.offset),(x.offset-y.offset));
    px4 = px3(1,:);
    x.data = [px4 x.data]; %new x.data
end

if x.offset<y.offset;
    py3 = zeros((y.offset-x.offset),(y.offset-x.offset));
    py4 = py3(1,:);
    y.data = [py4 y.data]; %new y.data
end

%concatenate zeros at the back of x.data or y.data if necessary

if (x.offset+length(pxo)-1)<(y.offset+length(pyo)-1);
    px5 = zeros((y.offset+length(pyo)-1)-(x.offset+length(pxo)-1),(y.offset+length(pyo)-1)-(x.offset+length(pxo)-1));
    px6 = px5(1,:);
    x.data = [x.data px6]; %new x.data
end

if (x.offset+length(pxo)-1)>(y.offset+length(pyo)-1);
    py5 = zeros(((x.offset+length(pxo)-1)-(y.offset+length(pyo)-1)),(x.offset+length(pxo)-1)-(y.offset+length(pyo)-1));
    py6 = py5(1,:);
    y.data = [y.data py6]; %new y.data
end

%add z.data = x.data + y.data
pz1=x.data + y.data;
```

```
%set z.offset
p10 = min(x.offset,y.offset);
p11 = find(pz1~=0,1,'first');
p12 = (p10+p11-1);

            %check if z.data is empty
            if all((pz1)==0);
              z = sequence(0,0);
                    return
            end

%remove zeros in front of, or at the back of z.data if necessary
while pz1(1)==0;
    pz1(1)=[];
end

pz1=fliplr(pz1);

while pz1(1)==0;
    pz1(1)=[];
end

pz1=fliplr(pz1);

%set z.data and z.offset into a defined sequence
z = sequence(pz1,p12);

    end

    function z = minus(x, y)

% MINUS  Subtract x and y. Either x and y will both be sequence structures, or one of them may be a number.
if isa(x,'sequence')==0;
p1 = sequence((x)*(ones(1,length(y.data))),y.offset);
x = p1;
end

if isa(y,'sequence')==0;
p2 = sequence((y)*(ones(1,length(x.data))),x.offset);
y = p2;
end

%concatenate zeros in front of x.data or y.data if necessary
pxo=x.data;
pyo=y.data;

if x.offset>y.offset;
    px3 = zeros((x.offset-y.offset),(x.offset-y.offset));
    px4 = px3(1,:);
    x.data = [px4 x.data]; %new x.data
end

if x.offset<y.offset;
    py3 = zeros((y.offset-x.offset),(y.offset-x.offset));
    py4 = py3(1,:);
    y.data = [py4 y.data]; %new y.data
end

%concatenate zeros at the back of x.data or y.data if necessary

if (x.offset+length(pxo)-1)<(y.offset+length(pyo)-1);
    px5 = zeros((y.offset+length(pyo)-1)-(x.offset+length(pxo)-1),(y.offset+length(pyo)-1)-(x.offset+length(pxo)-1));
    px6 = px5(1,:);
    x.data = [x.data px6]; %new x.data
end

if (x.offset+length(pxo)-1)>(y.offset+length(pyo)-1);
    py5 = zeros(((x.offset+length(pxo)-1)-(y.offset+length(pyo)-1)),(x.offset+length(pxo)-1)-(y.offset+length(pyo)-1));
    py6 = py5(1,:);
    y.data = [y.data py6]; %new y.data
end

%add z.data = x.data + y.data
pz1=x.data - y.data;

%set z.offset & eliminate leading zeros if necessary
p10 = min(x.offset,y.offset);
p11 = find(pz1~=0,1,'first');
p12 = (p10+p11-1);

            %check if z.data is empty
            if all((pz1)==0);
              z = sequence(0,0);
                    return
            end

%remove zeros in front of, or at the back of z.data if necessary
```

```
while pz1(1)==0;
    pz1(1)=[];
end

pz1=fliplr(pz1);

while pz1(1)==0;
    pz1(1)=[];
end

pz1=fliplr(pz1);

%set z.data and z.offset into a defined sequence
z = sequence(pz1,p12);

    end

    function z = times(x, y);
            % TIMES  Multiply x and y. Either x and y will both be sequence structures, or one of them may be a number.
if isa(x,'sequence')==0;
p1 = sequence((x)*(ones(1,length(y.data))),y.offset);
x = p1;
end

if isa(y,'sequence')==0;
p2 = sequence((y)*(ones(1,length(x.data))),x.offset);
y = p2;
end


%concatenate zeros in front of x.data or y.data if necessary
pxo=x.data;
pyo=y.data;


if x.offset>y.offset;
    px3 = zeros((x.offset-y.offset),(x.offset-y.offset));
    px4 = px3(1,:);
    x.data = [px4 x.data]; %new x.data
end

if x.offset<y.offset;
    py3 = zeros((y.offset-x.offset),(y.offset-x.offset));
    py4 = py3(1,:);
    y.data = [py4 y.data]; %new y.data
end

%concatenate zeros at the back of x.data or y.data if necessary

if (x.offset+length(pxo)-1)<(y.offset+length(pyo)-1);
    px5 = zeros((y.offset+length(pyo)-1)-(x.offset+length(pxo)-1),(y.offset+length(pyo)-1)-(x.offset+length(pxo)-1));
    px6 = px5(1,:);
    x.data = [x.data px6]; %new x.data
end

if (x.offset+length(pxo)-1)>(y.offset+length(pyo)-1);
    py5 = zeros(((x.offset+length(pxo)-1)-(y.offset+length(pyo)-1)),(x.offset+length(pxo)-1)-(y.offset+length(pyo)-1));
    py6 = py5(1,:);
    y.data = [y.data py6]; %new y.data
end

%add z.data = x.data + y.data
pz1=(x.data).*(y.data);

%set z.offset
p10 = min(x.offset,y.offset);
p11 = find(pz1~=0,1,'first');
p12 = (p10+p11-1);

            %check if z.data is empty
            if all((pz1)==0);
              z = sequence(0,0);
                    return
            end

%remove zeros in front of, or at the back of z.data if necessary
while pz1(1)==0;
    pz1(1)=[];
end

pz1=fliplr(pz1);

while pz1(1)==0;
    pz1(1)=[];
end

pz1=fliplr(pz1);
```

```
            %set z.data and z.offset into a defined sequence
            z = sequence(pz1,p12);

                end

                function stem(x);
                        % STEM Display a Matlab sequence, x, using a stem plot.
            st1=x.offset;
            st2=(length(x.data));
            st3=st1:st1+st2-1;
            stem(st3,x.data);
            xlim([(st1-1) (st1+st2)]);
            ylim([min(x.data)-1 max(x.data)+1]);
                end
        end
end
```



Published with MATLAB® R2018b